# ECM3412/ECMM409 Nature-Inspired Computation

## Introduction to the Module and Introduction to Evolutionary Algorithms

Dr Ed Keedwell
E.C.Keedwell@ex.ac.uk

# Module Information

- 15 Credits
- Assessment

| CA | Type of Task | Approx. Deadline | Worth |
|---|---|---|---|
| CA1 | Programming Task | ~22nd November | 20% |
| CA2 | Presentation | ~9th December | 10% |
| CA3 | Team Project (ECMM409 Only) | ~22nd January 09 | 70% |
| Exam | ECM3412 Only | May/June 09 | 70% |

- Study resources include lecture slides, module descriptors, suggested reading, CAs etc…
- http://www.secamlocal.ex.ac.uk/studyres/ECM3412/
- Also, examinable reading which I will discuss later in the module.

# Timetable Change

- This lecture will be moved from next week onwards to:
- Thursday at 3pm in 102
- Please let others know

# What is Nature-Inspired Computing?

- To understand this, we can split the title into its component parts:
  - "Nature" or what elements of nature are considered in NIC?
  - "Inspired" or how can nature influence computing?
  - "Computing" or what problems can be solved by NIC?

# “Nature”

- Which natural systems can benefit computing?
  - Evolution
  - Human-like activity (e.g. brain, immune system)
  - Collective behaviours (e.g. ant colonies, swarms)

# “Inspired”

- Why should natural systems be used as inspiration for new algorithms?
  - Evolution – created some of the most complex objects we know of – ourselves.
  - Human-like activity – we solve many seemingly difficult problems simultaneously
  - Collective behaviours – a collection of seemingly unintelligent individuals can display intelligence as part of a collective

# “Computing”

- What problems can be solved by using nature inspired computing?
  - Pattern recognition based loosely on
    - The workings of the brain
    - The human immune system
  - Problem solving based loosely on
    - Principles of evolution
    - The operation of ant colonies
    - Behaviour of swarms of individuals

# But why??

The nature-inspired techniques generally work much better than what came before!

E.g.

- Artificial neural networks are generally more successful than the classical `statistical' methods for pattern recognition.
- Artificial immune system methods are better at spotting potential threats than classical rule-based monitoring methods.
- Finally, evolutionary algorithms are much better than standard optimisation methods on a very large range of difficult problems.

# Syllabus

## Weeks 1—4 Evolutionary Algorithms

1-intro, 2-complexity, 3-problems & fitness landscapes, 4-genetic operators, 5-encodings/applications, 6-genetic programming, 7-multiobjective genetic algorithms, 8-EA workshop

## Week 5 - Reading Week (Probable)

## Weeks 6 and 7 Swarm Intelligence Methods

9-ant colony optimisation I, 10-ant colony II, 11-swarm intelligence I, 12-swarm intelligence II

## Weeks 8 and 9 Neural Networks

13-the human brain, 14-history of neural networks, 15-MLPs and gradient descent, 16 – using neural networks

## Week 10 Artificial Immune Systems & Artificial Life

17 & 18 Artificial Immune Systems, 19 & 20 Artificial Life (Cellular Automata)

## Revision Lecture (Closer to Exams)

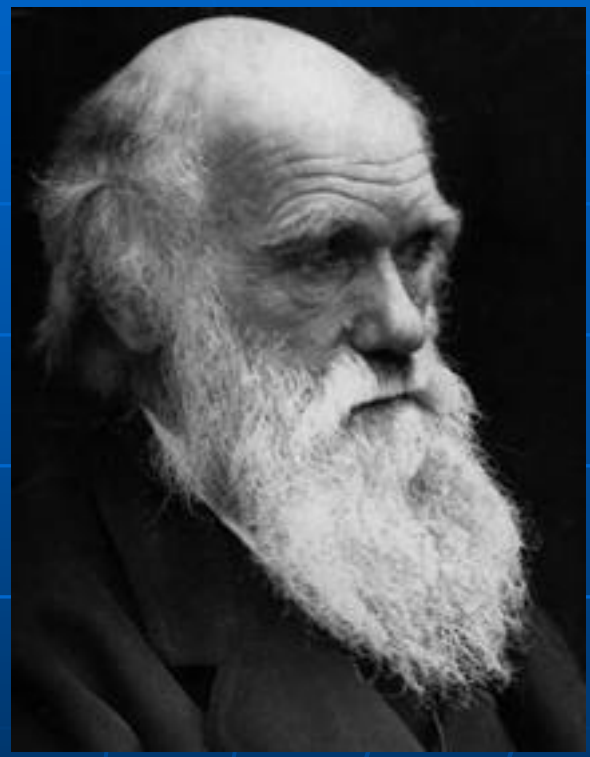
# Introduction to Evolutionary Computation

- Natural Evolution
- Search and Optimisation
- Hillclimbing / Gradient descent
- Local Search
- Population-Based Algorithms (i.e. Evolutionary Algorithms)
- Advantages and Disadvantages of EAs
- Applications of EAs
- Reading Material and Resources

# Natural Evolution

- “As many more individuals of each species are born than can possibly survive; and as, consequently, there is a frequently recurring struggle for existence, it follows that any being, if it vary however slightly in any manner profitable to itself, under the complex and sometimes varying conditions of life, will have a better chance of surviving, and thus be naturally selected. From the strong principle of inheritance, any selected variety will tend to propagate its new and modified form.”

*Charles Darwin, The Origin of Species (1859)*



# Evolution/Survival of the Fittest

In particular, any new mutation that appears in a child (e.g. longer neck, longer legs, thicker skin, longer gestation period, bigger brain, light-sensitive patch on the skin, a harmless `loose' bone, etc etc) and which *helps* it in its efforts to survive long enough to have children, will become more and more widespread in future generations.

The theory of evolution is the statement that all species on Earth have arisen in this way by evolution from one or more very simple self-reproducing molecules in the primeval soup. i.e. we have evolved via the accumulation of countless advantageous (in context) mutations over countless generations, and species have diversified to occupy niches, as a result of different environments favouring different mutations.

# Natural Evolution as a Problem Solving Method

We seem to have evolved from tiny stuff in the sea. How???

The theory is: given:

1. a population of organisms which have a lifetime and which can reproduce in a challenging/changing environment
2. a way of continually generating diversity in new `child' organisms

A `survival of the fittest principle will naturally emerge: organisms which tend to have healthy, fertile children will dominate (i.e. their descendents will).

# Evolution as Problem Solving

Here's a problem:

Design a material for the soles of boots that can help you walk up a smooth vertical brick wall

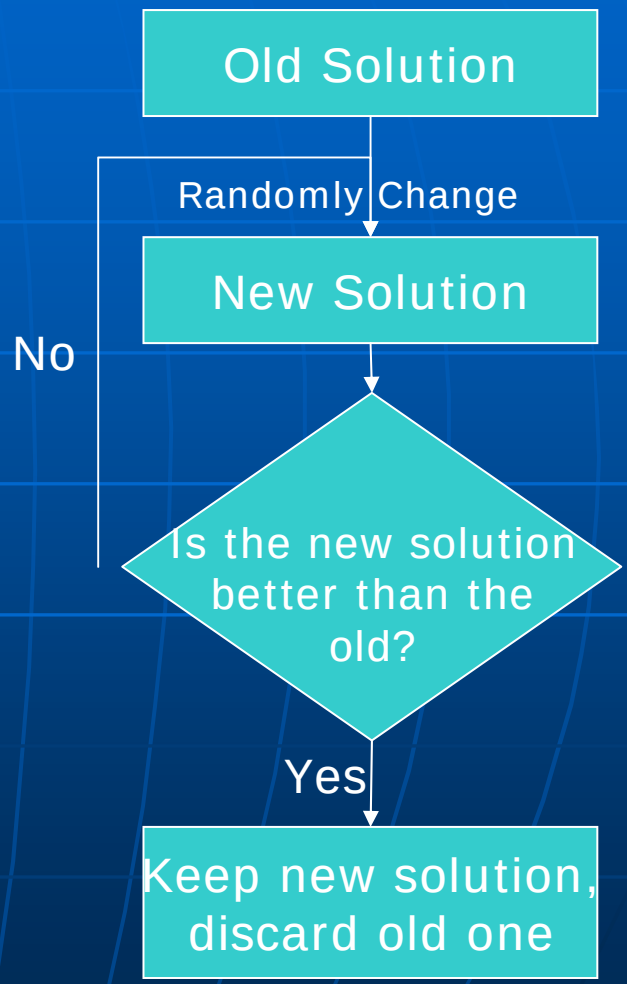
We haven't solved this, but nature has:
Geckos

# Evolution as a Problem Solving Method

Can view evolution as a way of solving the problem:

*How can I survive in this environment?*

The basic *method* of it is trial and error. I.e. evolution is in the family of methods that do something like this




But this appears to be a recipe for problem solving algorithms which take forever, with little or no eventual success!

# The Magic Ingredients

Not so – since there are **two vital** things (and one other sometimes useful thing) we learn from natural evolution, which, with a sprinkling of our own commonsense added, lead to generally superb problem solving methods called evolutionary algorithms:

Lesson1: Keep a population/collection of different things on the go.

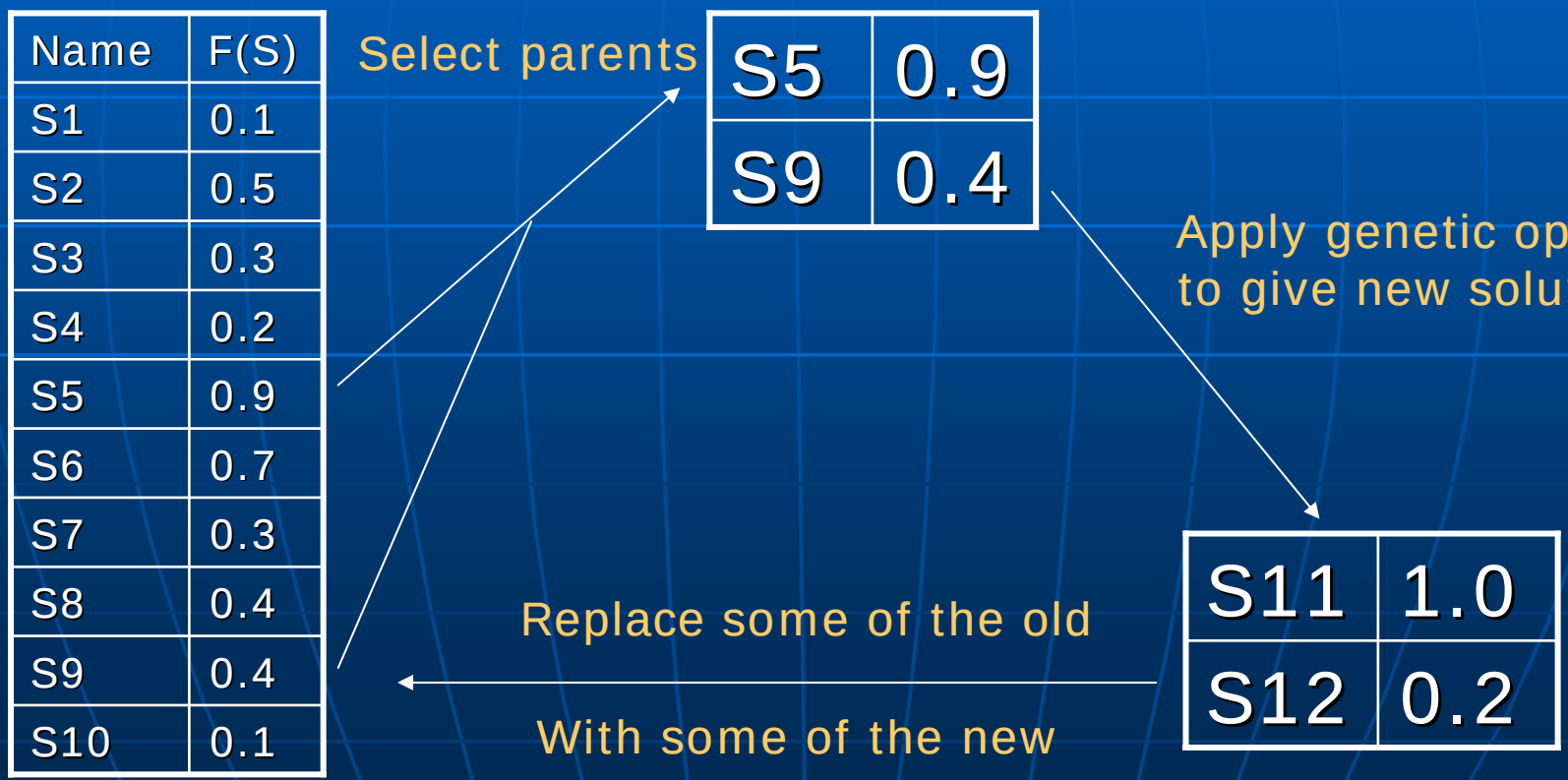
Lesson2: Select `parents' with a relatively *weak bias* towards the fittest.
It's not really plain survival of the fittest, what works is
*the fitter you are, the more chance you have to reproduce,*
and it works best if even the least fit still have some chance.

Lesson3: It can sometimes help to use *recombination* of two or more `parents' – I.e. generate new candidate solutions by combining bits and pieces from different previous solutions.

# A Generic Evolutionary Algorithm

Suppose you have to find a solution to some problem or other, and suppose, given any candidate solution *s* you have a function *f*(*s*) which measures how good *s* is as a solution to your problem.

Generate an initial population *P* of randomly generated solutions (this is typically 100 or 500 or so). Evaluate the fitness of each. Then go through 3 stages, Selection, Variation & Population Update



# Basic Varieties of Evolutionary Algorithm

1. Selection: Choose some of *P* to be *parents*

There are many different ways to select – e.g. choose top 10% of the population; choose with probability proportionate to fitness; choose randomly from top 20%, etc …

2. Variation: Apply genetic operators …

There are many different ways to do this, and it depends much on the encoding (see next slide). We will learn certain standard ways.

3. Population update: Update the population *P* by …

There are many several ways to do this, e.g. replace entire population with the new children; choose best |P| from *P* and the new ones, etc.

# Some of what EA-ists (theorists and practitioners) are Most concerned with:

***How to select***? Always select the best? Bad results, quickly
Select almost randomly? Great results, too slowly

***How to encode***? Can make all the difference, and is intricately tied up with :

***How to vary***? (mutation, recombination, etc…)
small-step mutation preferred, recombination seems to be a principled way to do large steps, but large steps are usually abysmal.

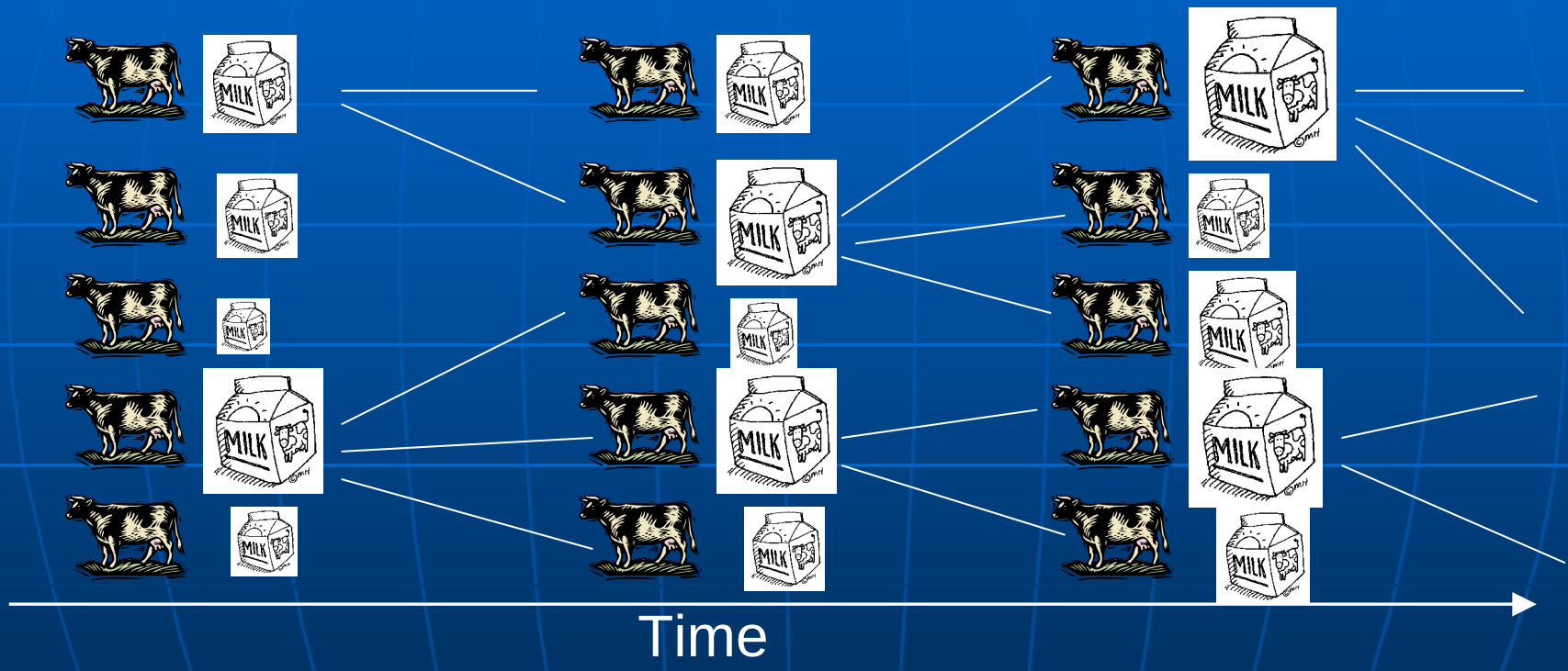
***What parameters***? ***How to adapt with time***?

# What are they good for ?

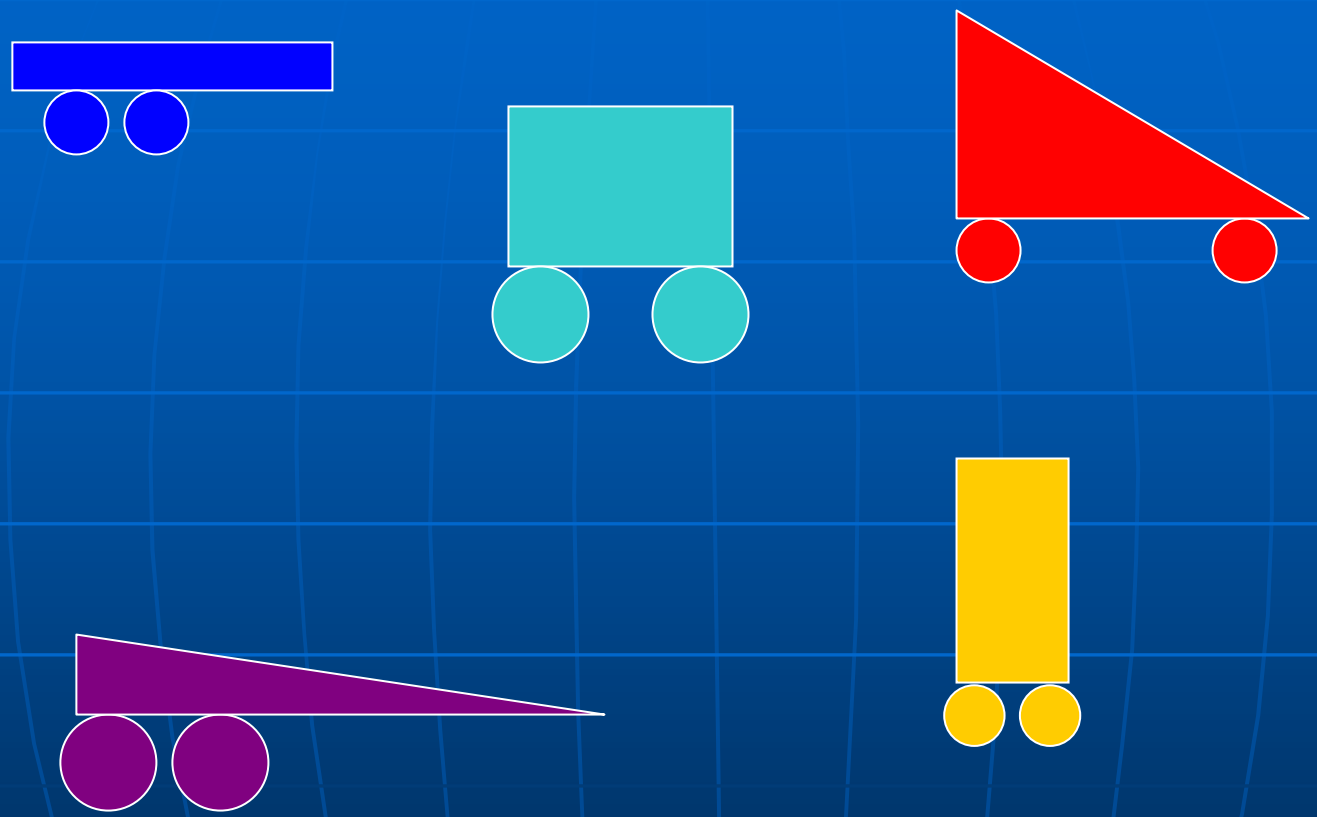
Suppose we want the best possible schedule for a **university lecture timetable**.

- Or the best possible **pipe network design** for delivering water
- Or the best possible design for an **antenna** with given requirements
- Or a **formula** that fits a curve better than any others
- Or the best placement strategy for **mobile phone masts**
- Or the most efficient **orbits for satellites**
- Or the best **strategy for flying a fighter aircraft**
- Or the most accurate **neural network** for a data mining or control problem,
- Or the best **treatment plan** (beam shapes and angles) for radiotherapy cancer treatment
- And so on and so on ….!
- The applications cover all of optimisation and machine learning.

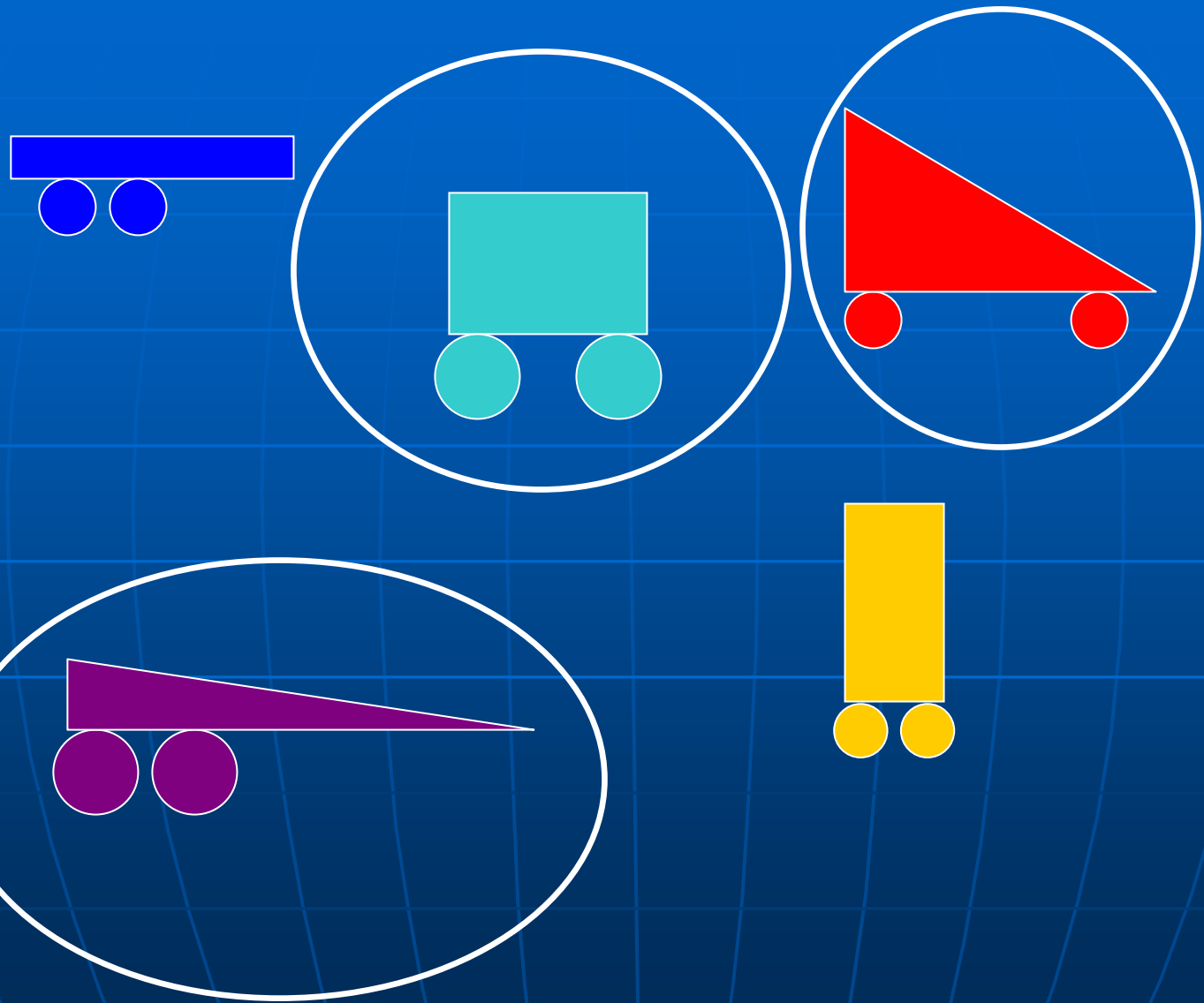
# More like selective breeding than natural evolution



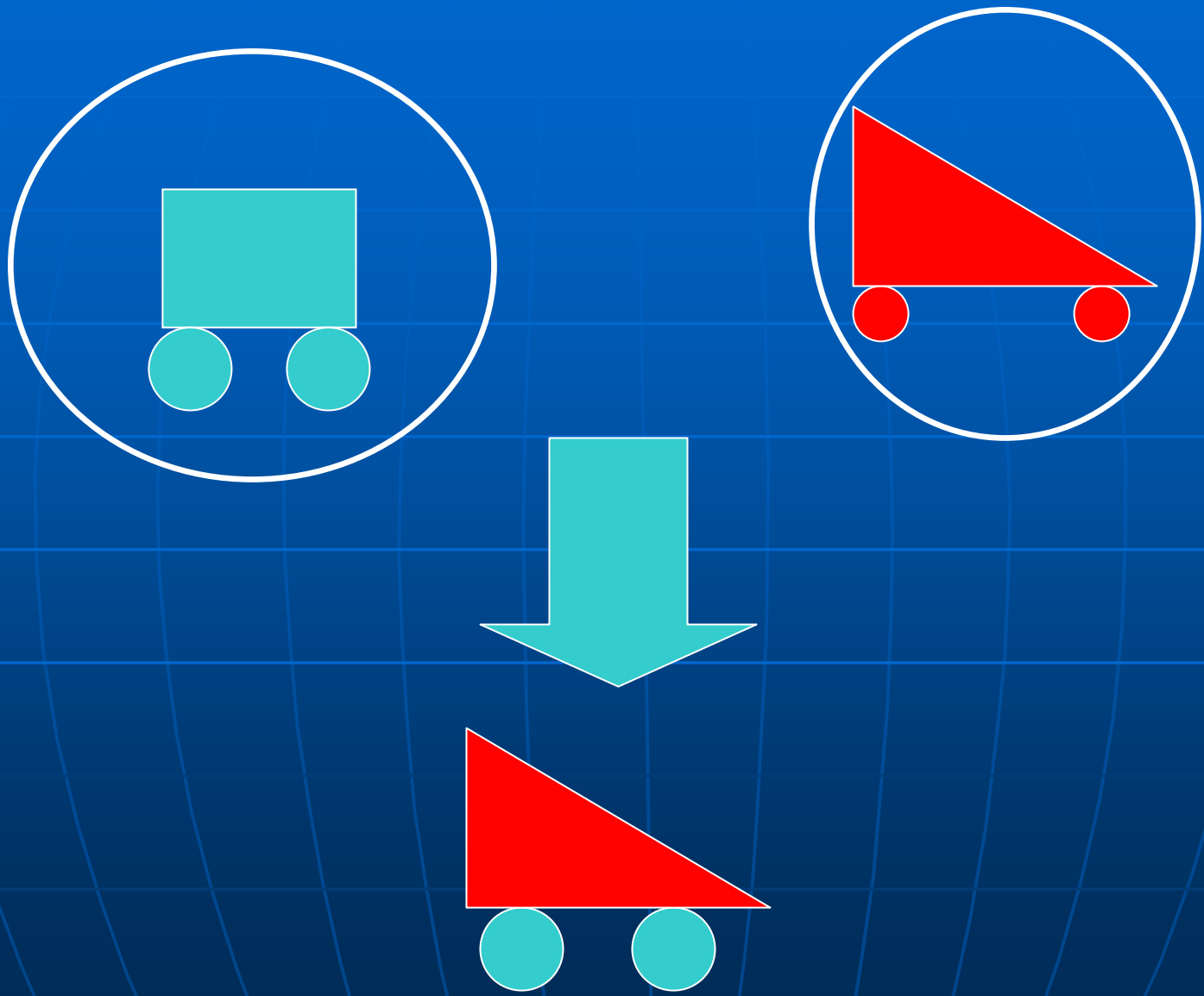
# Initial population



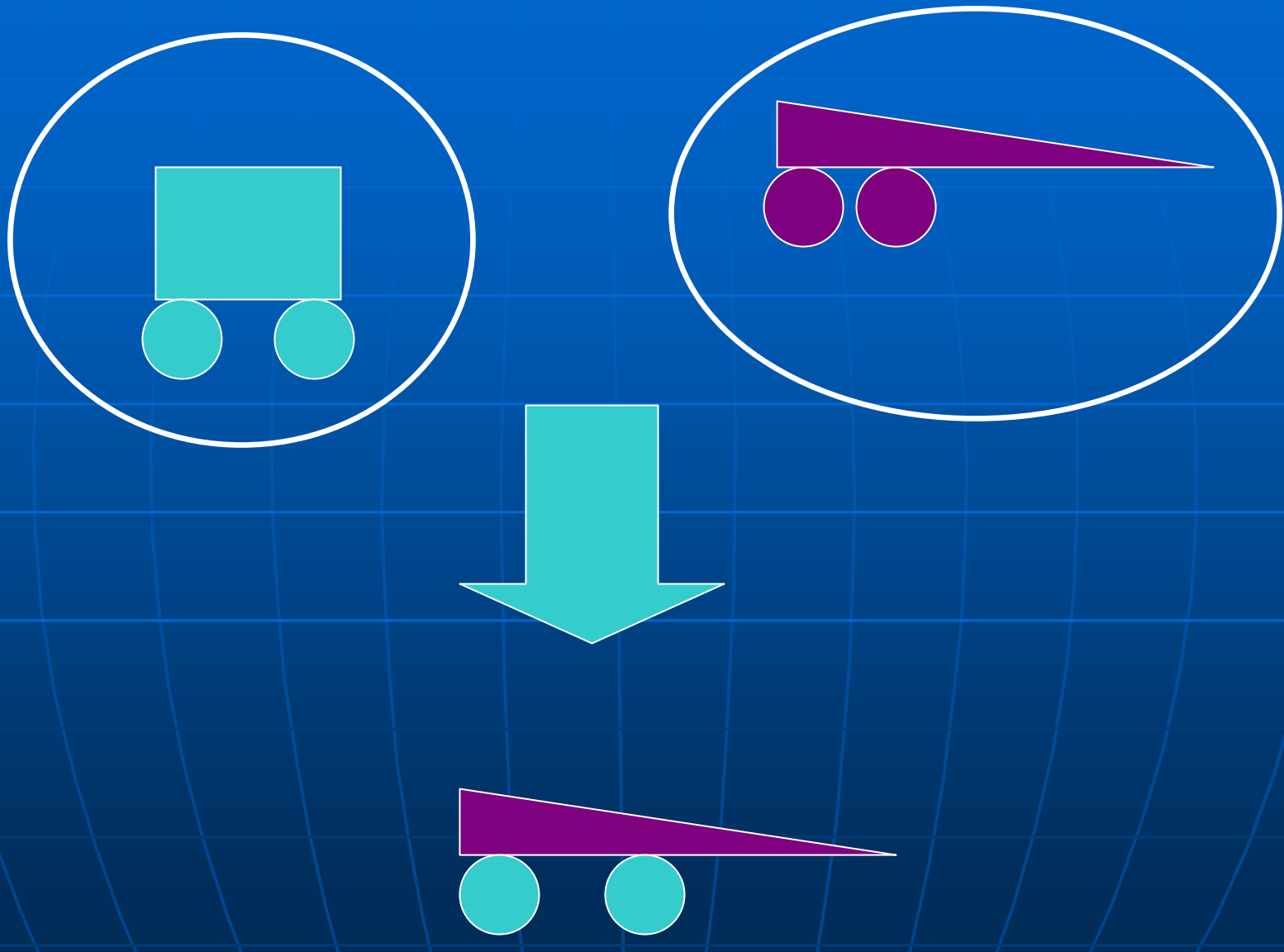
# Select



# Crossover



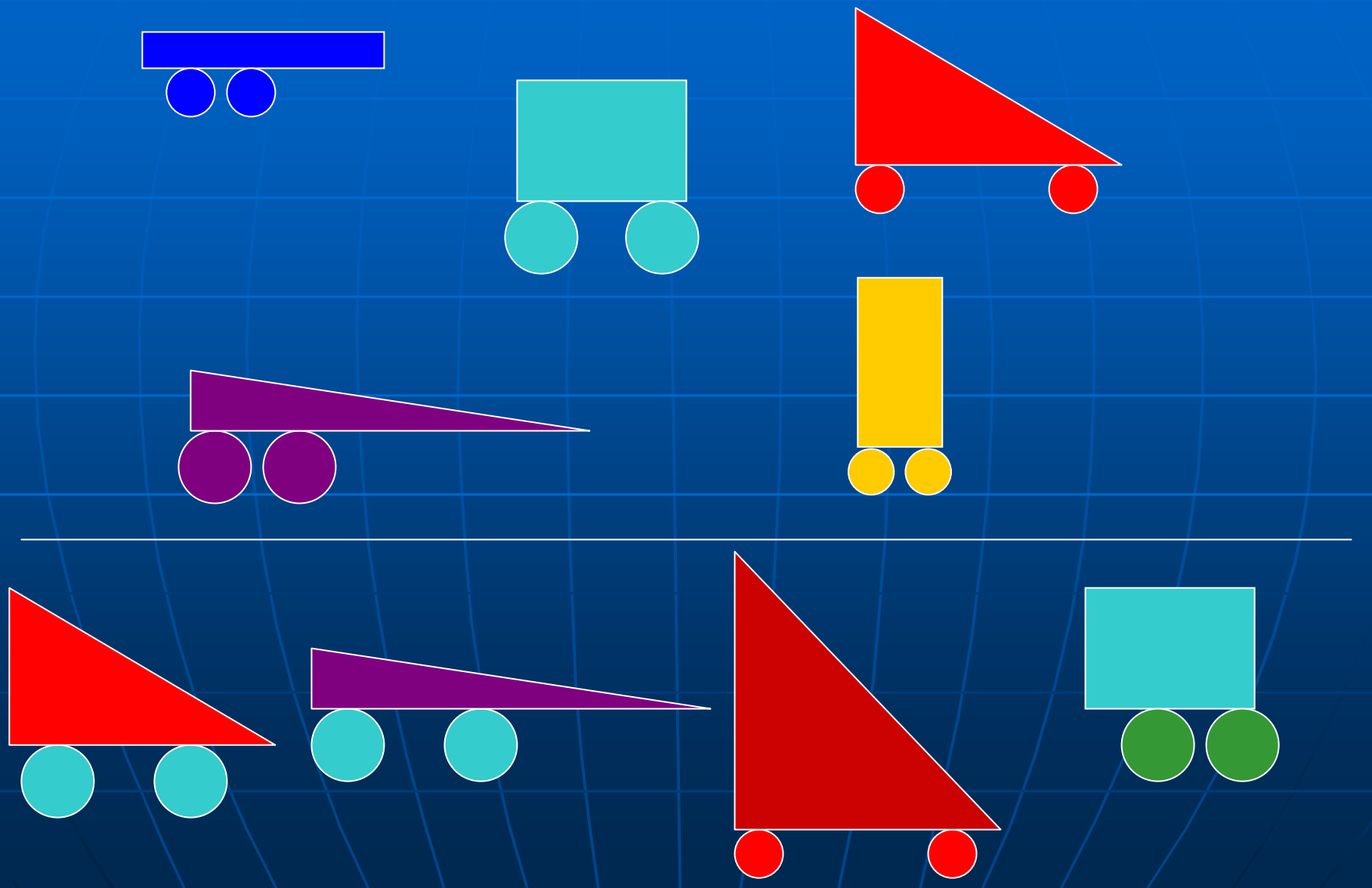
# Another Crossover



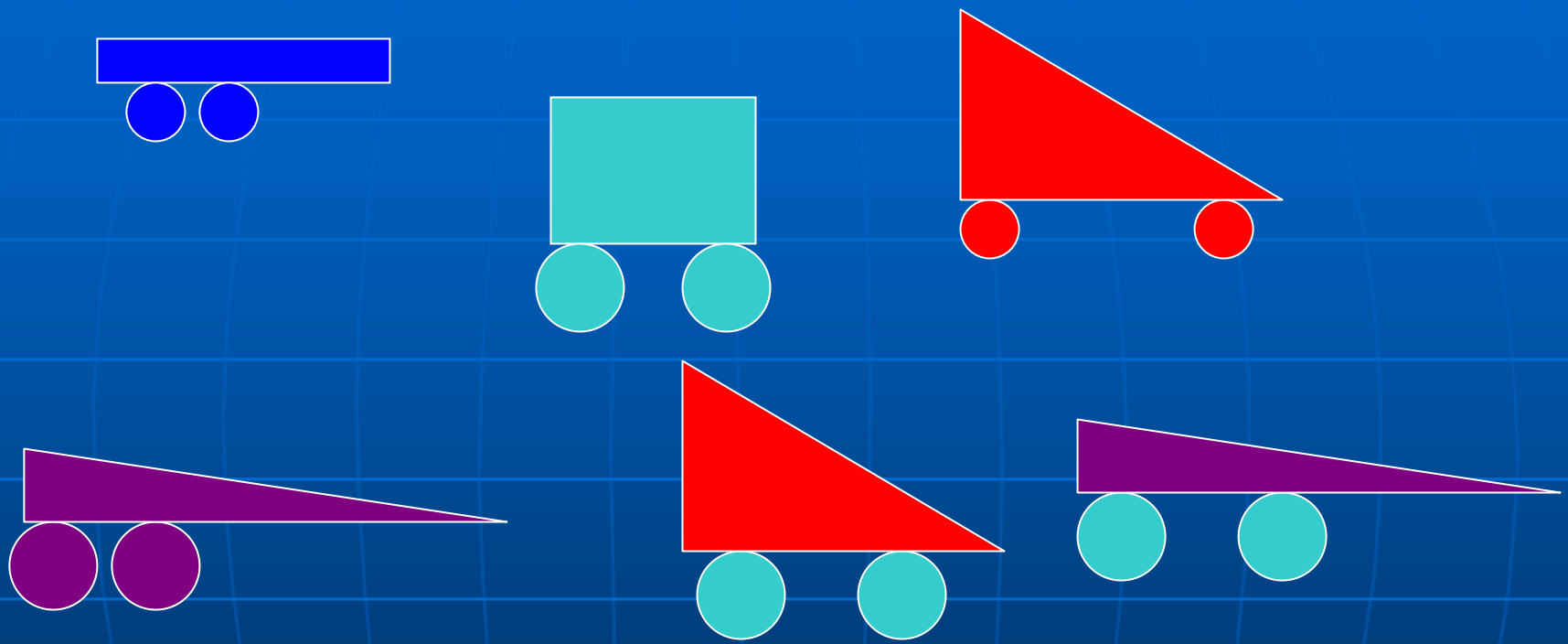
# A mutation

# Another Mutation

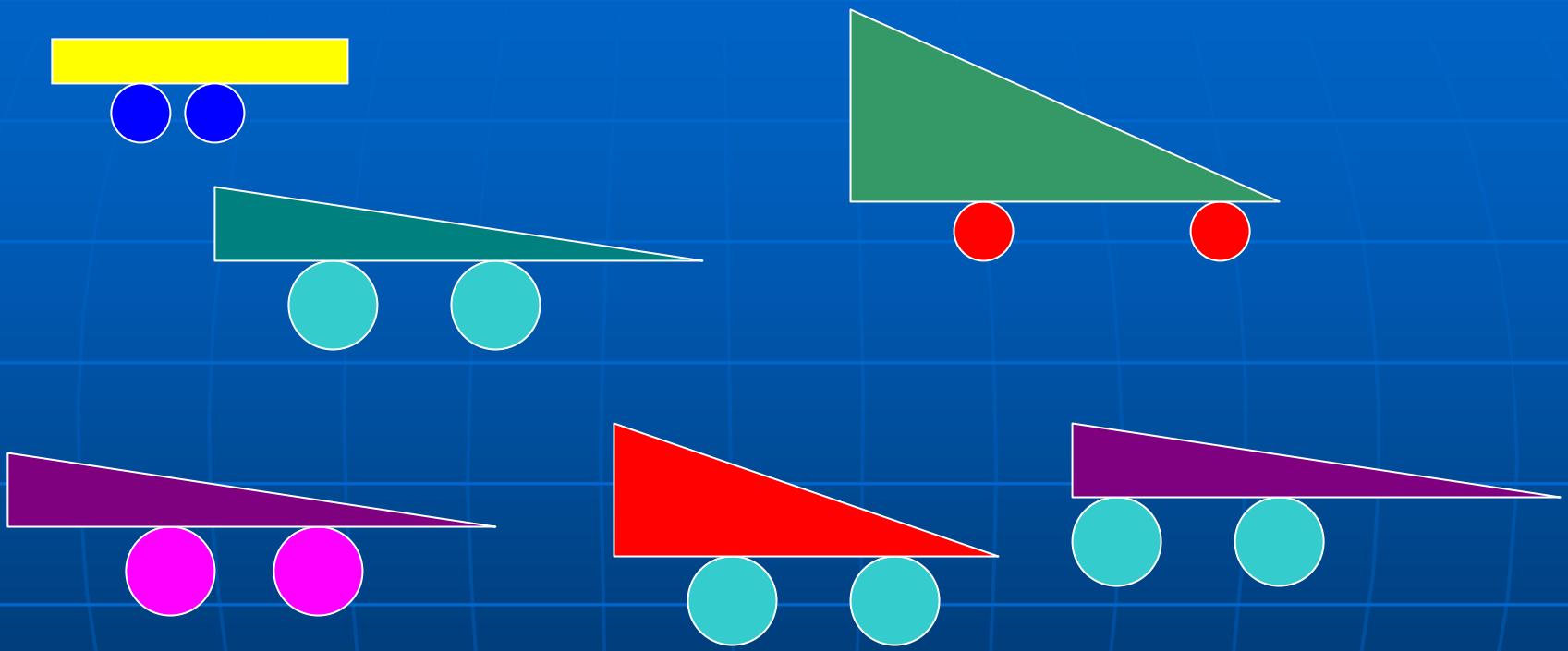
# Old population + children



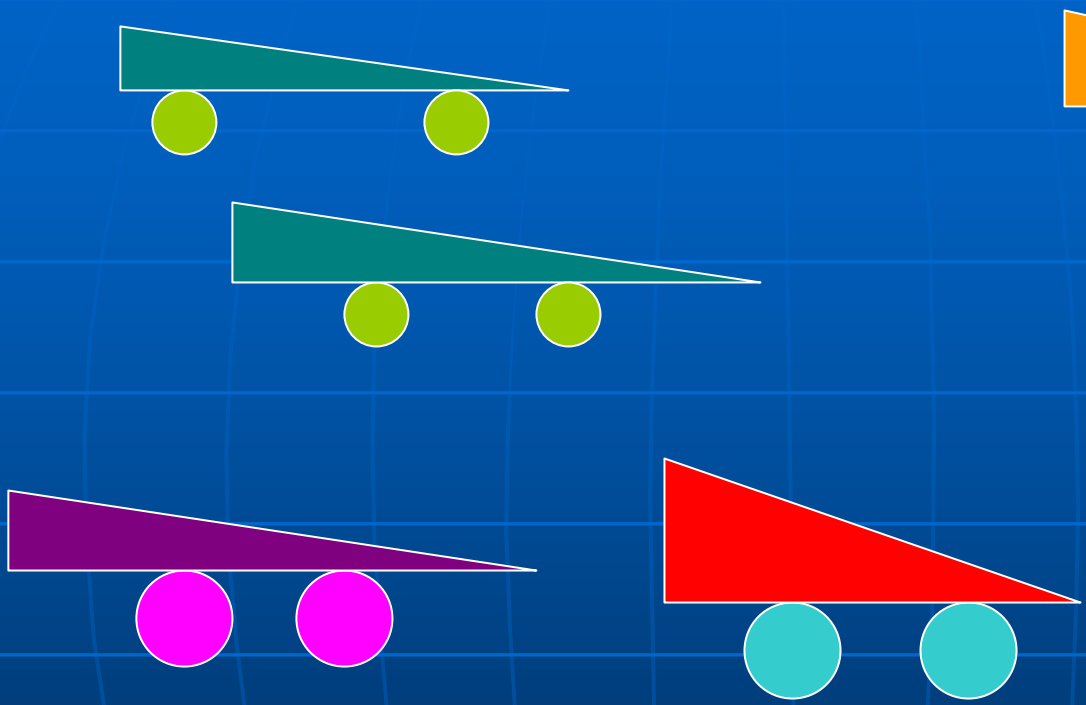
# New Population: Generation 2



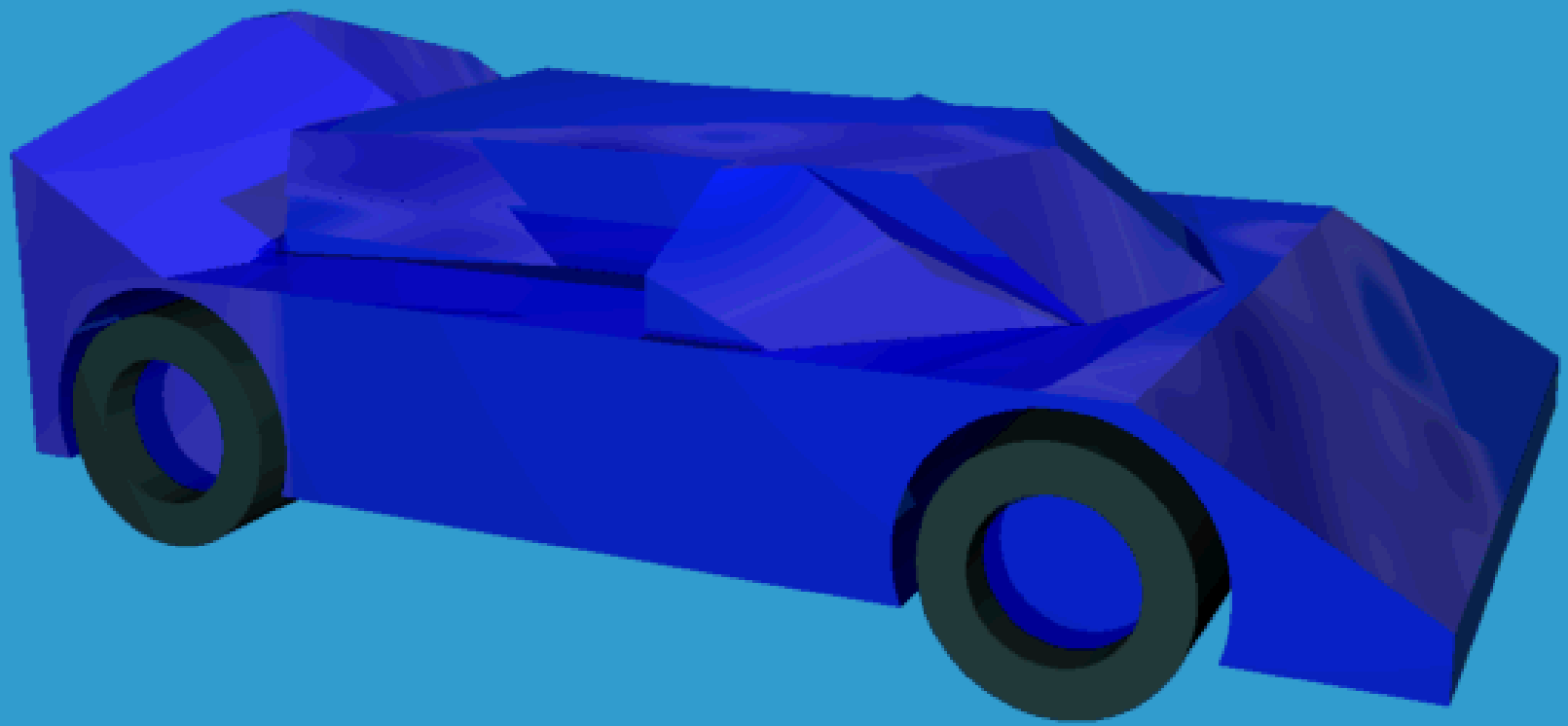
# Generation 3



# Generation 4, etc …



Bentley’s thesis work

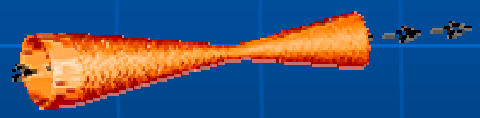


Fixed wheel positions, constrained bounding area,
Chromosome is a series of  slices
\fitnesses evaluated via a simple airflow simulation

One of the very first applications. Determine the internal shape of a two-phase jet nozzle that can achieve the maximum possible thrust under given starting conditions

Ingo Rechenberg was the *very* first, with pipe-bend design

This is slightly later work in the same lab, by Schwefel

Starting point 

EA (ES) running

Result 

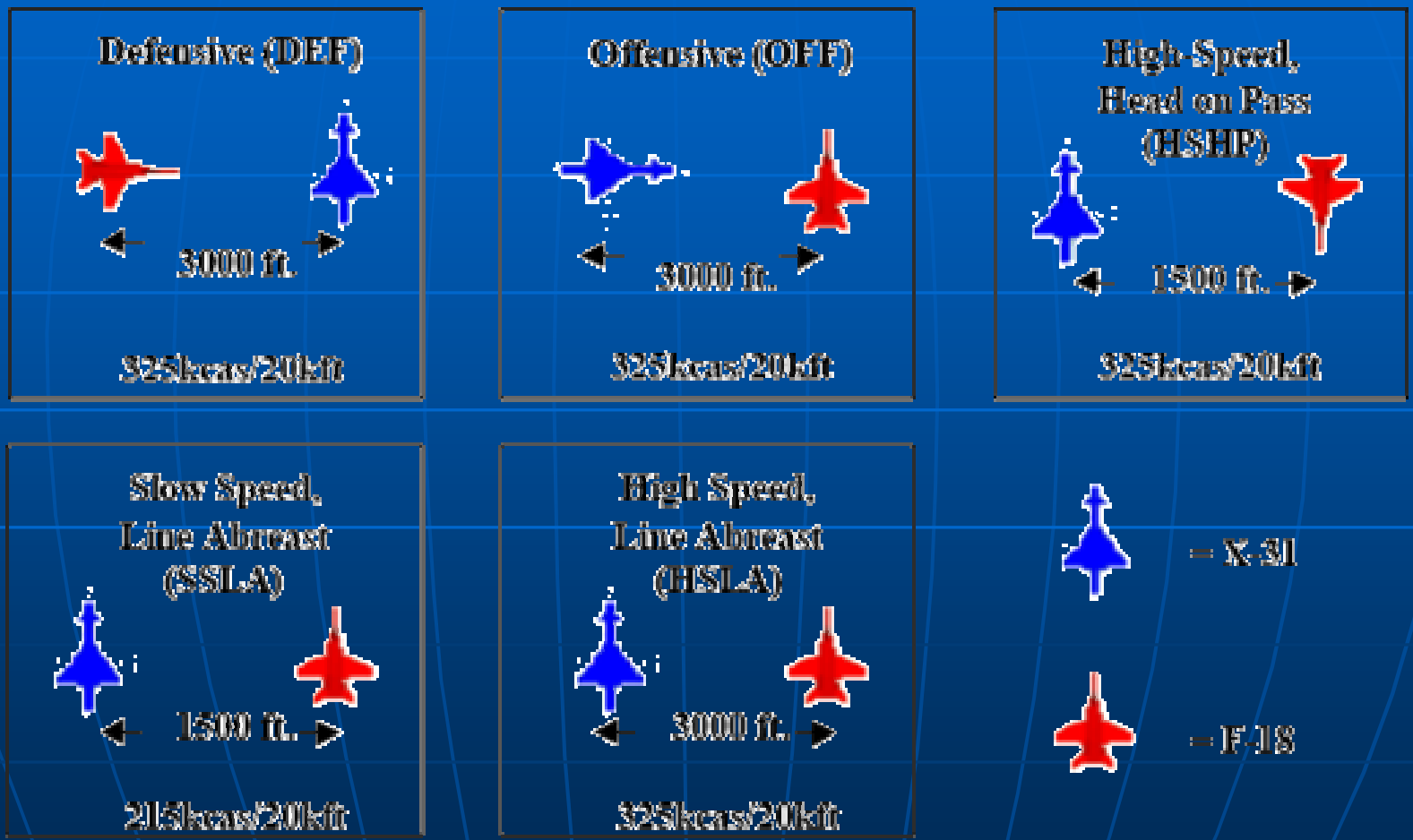
A recurring theme: design freedom → entirely new and better designs based on principles we don't yet understand.

# Optimising Formula One Car Setups

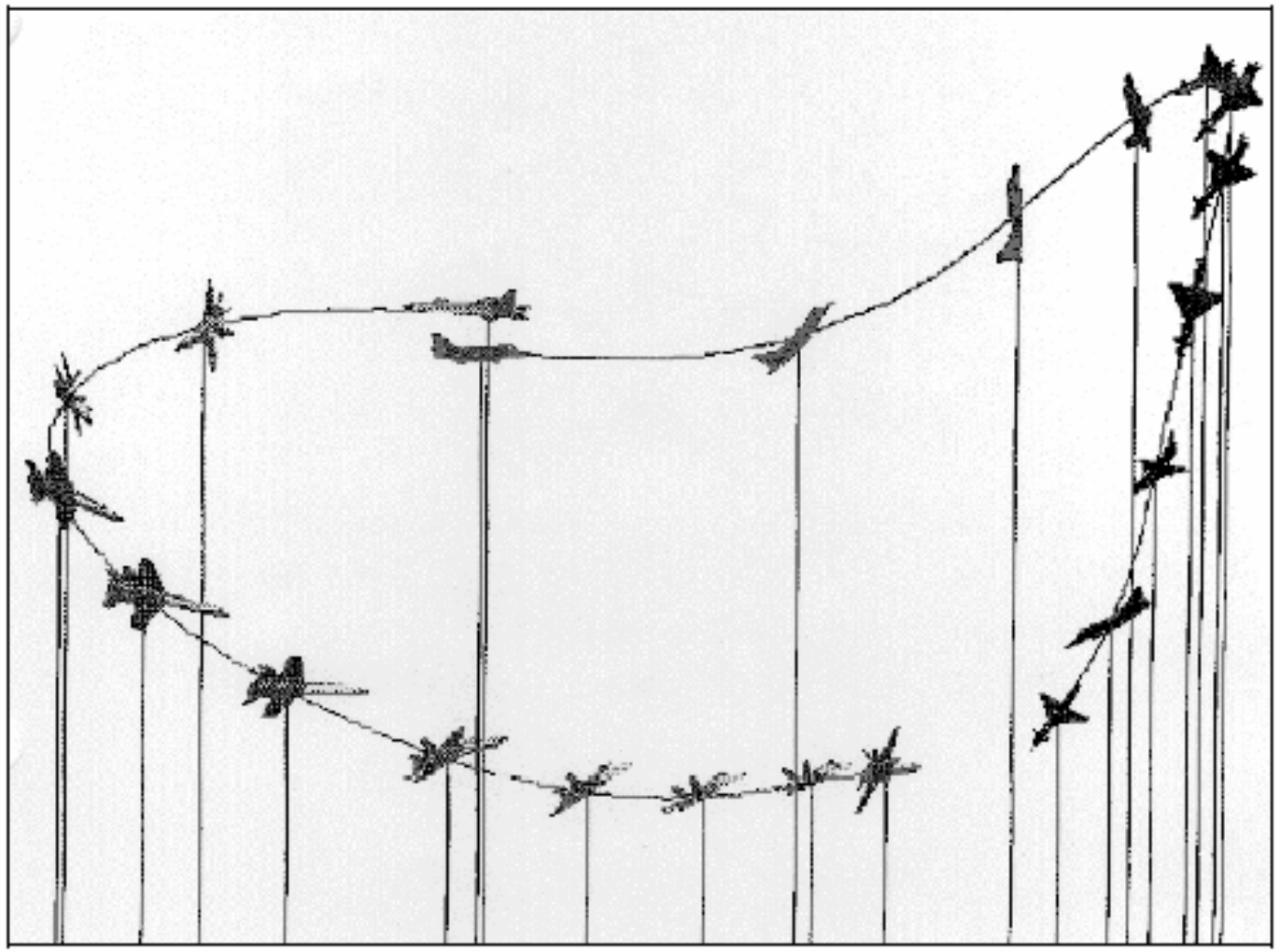
Wloch and Bentley (2004) used an EA and a PC Formula One Game/Simulator to optimise the setup of the car (e.g. suspension, tyres, brakes etc…)

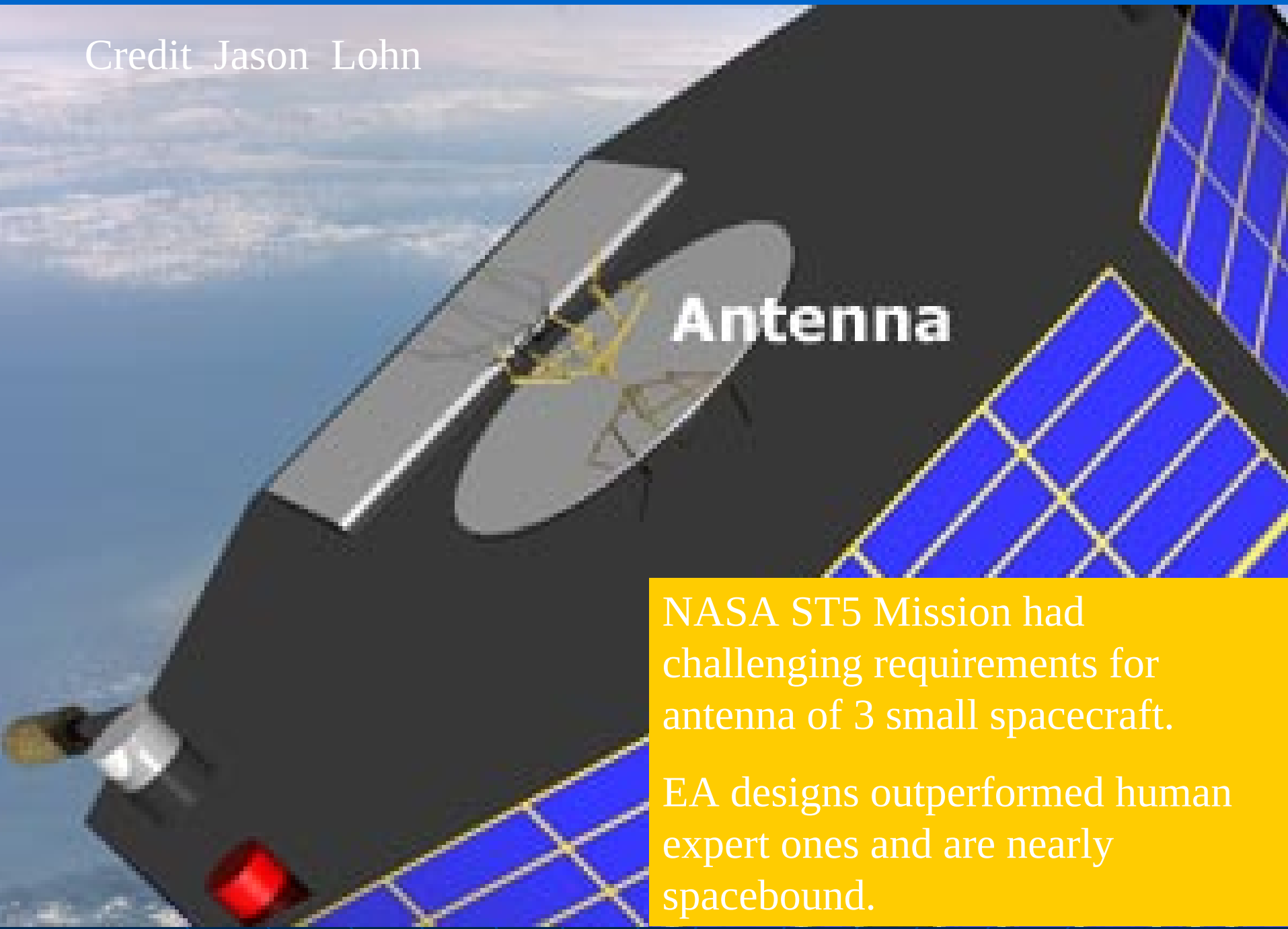
**Figure 3.** Image produced by the racing simulator, *Electronic Arts™ Formula One Challenge '99-'02*. The speed of the Formula One car had to drop from over 170 mph, on entry to this corner, to 50 mph. With settings evolved in experiment 3, the front left wheel locks. This problem did not occur in the setups produced in experiment 4.

# Evolving Top Gun strategies



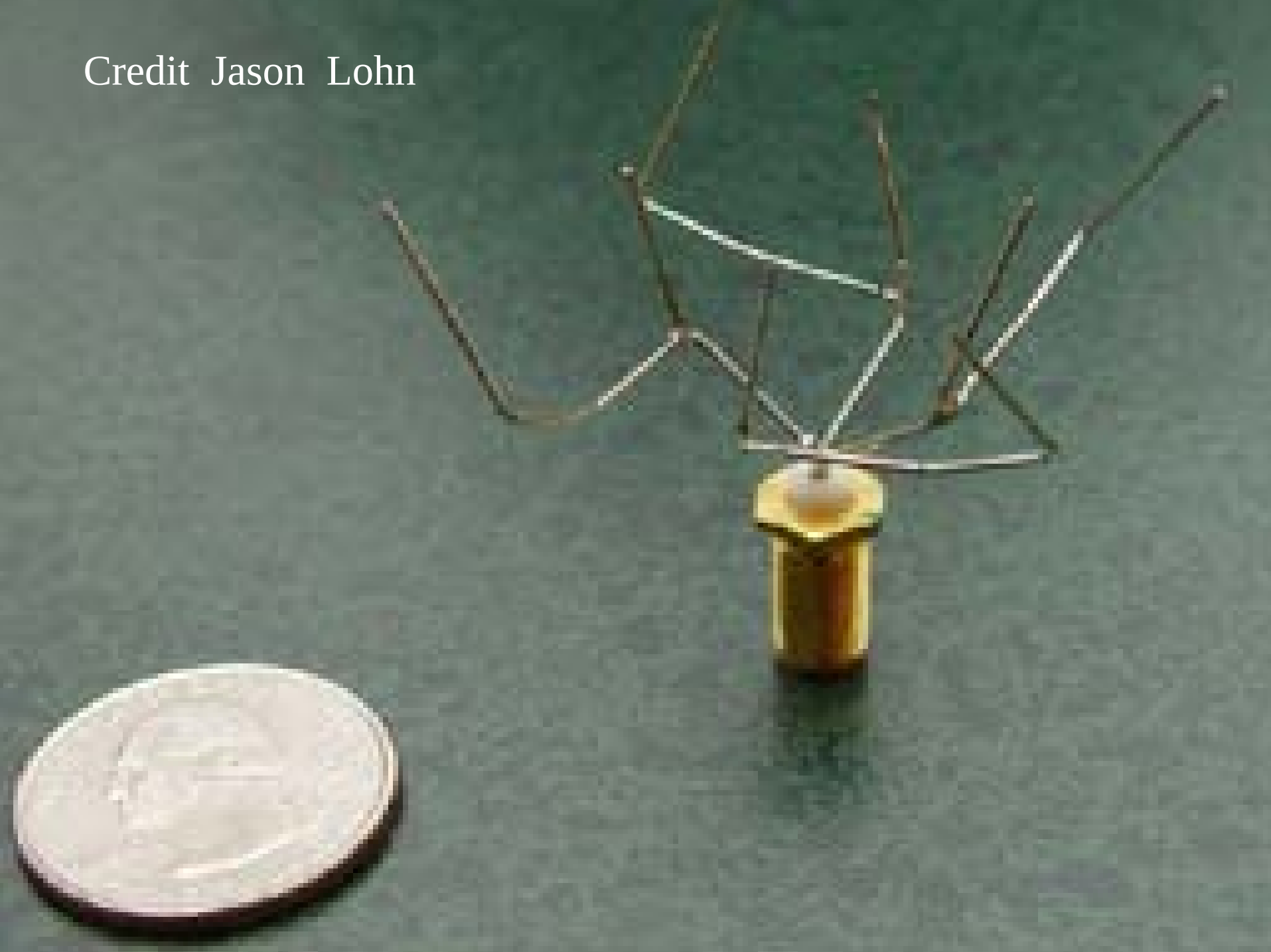
# Evolving Top Gun strategies






NASA ST5 Mission had challenging requirements for antenna of 3 small spacecraft.

EA designs outperformed human expert ones and are nearly spacebound.


Credit Jason Lohn